ODE: An Online Data Selection Framework for Federated Learning With Limited Storage

Chen Gong, *Student Member, IEEE*, Zhenzhe Zheng, *Member, IEEE, ACM*, Yunfeng Shao, Bingshuai Li, Fan Wu, *Member, IEEE*, and Guihai Chen, *Fellow, IEEE*

***Abstract*— Machine learning (ML) models have been deployed in mobile networks to deal with massive data from different layers to enable automated network management. To overcome high communication cost and severe privacy concerns of centralized ML, federated learning (FL) has been proposed to achieve distributed ML among numerous networked devices. While the computation and communication limitation has been widely studied, the impact of limited storage of mobile devices on the performance of FL is still not explored. Without an effective data selection policy to filter the massive streaming networked data on devices, classical FL can suffer from much longer model training time ($4\times$) and dramatic inference accuracy reduction (7%), observed in our experiments. In this work, we take the first step to consider the online data selection for FL with limited on-device storage. We first define a new data valuation metric for data selection in FL with theoretical guarantee for simultaneously accelerating model convergence and enhancing final accuracy. We further design ODE, an Online Data sElection framework for FL, to coordinate networked devices to store valuable data samples collaboratively. Experimental results on one industrial dataset and three public datasets show the remarkable advantages of ODE over the state-of-the-art approaches. Particularly, on the industrial dataset, ODE achieves as high as $2.5\times$ speedup of training time and 6% increase in final accuracy, and is robust to various factors in practical environments.**

***Index Terms*— Federated learning, limited on-device storage, online data selection.**

I. Introduction

THE next-generation mobile computing systems require effective and efficient management of mobile networks and devices in various aspects, including resource provisioning [1], security and intrusion detection [2], quality of service guarantee [3], and etc. Analyzing and controlling such an increasingly complex mobile network with traditional human-in-the-loop approaches will not be possible any more, due to low-latency requirement, massive real-time data and complicated correlation among data [4], [5]. For example, in network traffic analysis, a fundamental task in mobile networks, devices such as routers and ONTs (Optical Network Terminal) can receive as many as $5,000$ packets per second. It is impractical to manually analyze massive high-dimensional data within milliseconds. Thus, machine learning (ML) models have been widely applied to discover pattern behind high-dimensional networked data, enable data-driven network control, and fully automate the mobile network operation [6].

Despite that ML model overcomes the limitation of human-in-the-loop approaches, its good performance highly relies on the abundant high quality data for model training [7], which is hard to obtain in mobile networks as the data is resided on heterogeneous devices in a distributed manner. On one hand, a ML model trained locally with limited data and computational resources is unlikely to achieve desirable inference accuracy and generalization ability [8]. On the other hand, directly transmitting data from distributed networked devices to a cloud server for centralized learning (CL) will bring prohibitively high communication cost and severe privacy concerns. Recently, federated learning (FL) [9] emerges as a distributed privacy-preserving ML paradigm to resolve the above concerns, which allows networked devices to upload local model updates instead of raw data, and a central server to aggregate these local models into a global model, solving the on-device data limitation, communication bottlenecks and potential privacy leakage.

Motivation and New Problem. For applying FL to mobile networks, we identify two unique properties of networked devices: limited on-device storage and streaming networked data, which have not been fully considered in previous FL literature. (i) Limited on-device storage: due to hardware resource constraints, mobile devices have restricted storage space for each mobile application, and can reserve only a small space to store data samples for model training without compromising the quality of other services. For example, most smart home routers have only 9-32MB storage to support various kinds of services [10], and thus only tens of training data samples can be stored in this scenario. (ii) Streaming networked data: data samples are continuously collected by mobile devices in a streaming manner, which need to make online decisions on whether to store each arrived data sample.

Manuscript received 11 May 2023; revised 18 November 2023; accepted 15 January 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Pedarsani. Date of publication 26 March 2024; date of current version 20 August 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022ZD0119100; and in part by China NSF under Grant 62322206, Grant 62132018, Grant U2268204, Grant 62025204, Grant 62272307, and Grant 62372296. *(Corresponding author: Zhenzhe Zheng.)*

Chen Gong, Zhenzhe Zheng, Fan Wu, and Guihai Chen are with Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: gongchen@sjtu.edu.cn; zhengzhenzhe@sjtu.edu.cn; fwu@cs.sjtu.edu.cn; gchen@cs.sjtu.edu.cn).

Yunfeng Shao and Bingshuai Li are with the Huawei Noah's Ark Laboratory, Beijing 100085, China (e-mail: shaoyunfeng@huawei.com; libingshuai@huawei.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2024.3365534, provided by the authors.

Digital Object Identifier 10.1109/TNET.2024.3365534

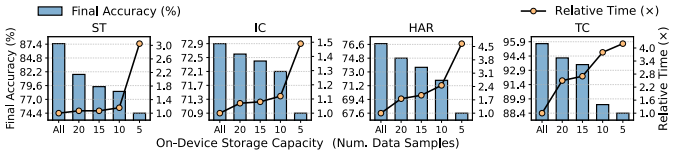


Fig. 1. Motivating experiments on 4 classic tasks to demonstrate the severe impact of limited on-device storage on final accuracy and training time of FL. (ST: synthetic task, IC: image classification, HAR: human activity recognition, TC: network traffic classification.)

Without a carefully designed data selection policy to maintain the data samples in storage, the empirical distribution of stored data could deviate from the true data distribution, which further complicates the notorious problem of not independent and identically distributed (non-iid) data distribution in FL [11], [12]. Our experiments on a wide range of FL tasks reveal that the traditional random selection policy significantly degrades the performance of classic FL process in both model training time and final inference accuracy, with $4\times$ longer training time and 7% accuracy reduction on network traffic classification task, $3.05\times$ and 13.1% on synthetic task, $1.49\times$ and 2.1% on image classification task, and $4.65\times$ and 9.1% on human activity recognition task (Figure 1). This is unacceptable in modern mobile networks, because the longer training time diminishes the timeliness of ML models in dynamic environments, and accuracy reduction results in failure to guarantee the quality of service [3]. Therefore, a fundamental problem when applying FL to mobile networks is how to select valuable data samples from on-device streaming networked data to simultaneously accelerate model training convergence and enhance final inference accuracy?

Design Challenges. The design of such an online data selection framework for FL involves three key challenges:

First, there is still no theoretical understanding about the impact of local on-device data on the training speedup and accuracy enhancement of global model in FL. Lacking information about raw data and local models of the other devices, it is challenging for one device to individually figure out the impact of its local data sample on the performance of the global model. Furthermore, the data sample-level correlation between convergence rate and model accuracy is still not explored in FL, and it is non-trivial to simultaneously improve these two aspects through one unified data valuation metric.

Second, the lack of temporal and spatial information complicates the online data selection in FL. For streaming networked data, we could not access the data samples coming from the future or discarded in the past. Lacking such temporal information, one device is not able to leverage complete statistical information (e.g. unbiased local data distribution) for accurate data valuation like outliers and noise detection [13], [14]. Further, due to the distributed paradigm of FL, one device cannot conduct effective data selection without the knowledge of other devices' stored data and local models, denoted as spatial information. This is because the valuable data samples selected by each device may overlap with the data selected by other devices. As a result, the locally valuable data may not be globally valuable.

Third, data selection on mobile devices needs to be low computation-and-memory cost due to the conflict between limited hardware resources and requirement on quality of user experience. As the additional time delay and memory costs introduced by online data selection process would degrade the performance of mobile network and user experience, the real-time data samples must be evaluated efficiently. However, the increasingly complex ML models lead to high computational complexity as well as large memory footprint for storing intermediate model outputs during data selection.

Limitations of Related Works. The prior works on data evaluation and selection failed to solve the above challenges. (i) Data selection methods in CL, such as leave-one-out [15], data shapley [16] and importance sampling [17], are not appropriate for FL due to the first challenge: they could only measure the value of each data sample corresponding to the local model training process, rather than the global model in FL. (ii) Data selection methods in FL did not consider the two new properties of FL devices. Mercury [8], FedBalancer [13] and the work from Li et al. [14] adopted importance sampling [17] to select the data samples with high loss or large gradient norm for training time speedup, but failed to solve the second challenge: these methods need to leverage the whole dataset or complete data distribution to normalize the sampling weight and remove the outliers and noise.

Our Solutions. To solve the above challenges, we design ODE, an online data selection framework that coordinates networked devices to select and store valuable data samples locally and collaboratively in FL, with theoretical guarantees for simultaneously accelerating model convergence and enhancing inference accuracy.

In ODE, we first theoretically analyze the impact of an individual local data sample on the convergence rate and final accuracy of the global model in FL. We discover a common dominant term in these two analytical expressions, *i.e.*, $\langle \nabla_w l(w, x, y), \nabla_w F(w) \rangle$, which implies that the projection of the gradient of a local data sample to the global gradient is a reasonable metric for data selection in FL. As this metric is a deterministic value, each device can simply maintain a priority queue to store the valuable data samples. Second, considering the lack of temporal and spatial information, we propose an efficient method for clients to approximate this data selection metric by maintaining a local gradient estimator on each device and a global one on the server. Third, to overcome the potential overlap of the stored data caused by data selection in a distributed manner, we further propose a strategy for the server to coordinate each device to store valuable data from different data distribution regions. Fourth, to achieve computation and memory efficiency, we propose a simplified version of ODE, which replaces the full model gradient with partial model gradient to concurrently reduce the computational cost of model backpropagation and save the memory for buffering intermediate outputs of ML models.

Implementation and Evaluation. We evaluated ODE on three public tasks: synthetic task (ST) [18], Image Classification (IC) [19] and Human Activity Recognition (HAR) [20], as well as one industrial traffic classification dataset (TC) collected from our 30-days deployment on 30 ONTs in practice, consisting of $560,000+$ packets from 250 mobile applications. We compare ODE against three categories of

data selection methods: random sampling [21], classic data sampling approaches in CL [8], [13], [14] and data selection methods for FL [13], [14]. The experimental results show that ODE outperforms all these baselines, achieving as high as 9.52× speedup of model training and 7.56% increase in final model accuracy on ST, 1.35× and 1.4% on IC, 2.22× and 6.38% on HAR, 2.5× and 6% on TC, with marginal extra time delay and memory footprint. Moreover, ODE is robust to different environmental factors, including local training epoch number, client participation rate, on-device storage capacity, mini-batch size and data heterogeneity across devices. We also conduct ablation experiments to demonstrate the effectiveness of each key component in ODE.

Summary of Contributions. Our contributions in this work can be summarized as follows: (i) To the best of our knowledge, we are the first to identify two new properties of mobile devices when applying FL to mobile networks, limited on-device storage and streaming networked data, and demonstrate their enormity on FL process. (ii) We provide analytical formulas on the impact of an individual local data sample on the convergence rate and the final inference accuracy of the global model, based on which we propose a new data valuation metric for data selection in FL with theoretical guarantee for simultaneously accelerating model convergence and improving inference accuracy. Furthermore, we propose ODE, an online data selection framework for FL, to realize on-device data selection and cross-device collaborative data storage. (iii) We conduct extensive experiments on three public datasets and one industrial traffic classification dataset to demonstrate the remarkable advantages of ODE against existing methods in FL.

II. Preliminaries

In this section, we present the training process of FL on mobile device. We consider the synchronous FL framework [9], where a server coordinates a set of mobile devices/clients[1] C to conduct distributed model training. Each client $c \in C$ generates data samples in a streaming manner with a velocity v_c. We use P_c to denote the client c's underlying distribution of local data, and $\tilde{P}_c$ to represent the empirical distribution of the stored data, denoted as B_c. The goal of FL is to train a global model w from the locally stored data $\tilde{P} = \bigcup_{c\in C} \tilde{P}_c$ with good performance with respect to the underlying distribution of overall data $P = \bigcup_{c\in C} P_c$:

$$\min_w F(w) = \sum_{c\in C} \zeta_c \cdot F_c(w),$$

where $\zeta_c = \frac{v_c}{\sum_{c'\in C} v_{c'}}$ denotes the normalized weight of each client, $F_c(w) = \mathbb{E}_{(x,y)\sim P_c}\big[l(w,x,y)\big]$ is the expected loss of the model w over the underlying data distribution of client c. We also use $\tilde{F}_c(w) = \frac{1}{|B_c|}\sum_{x,y\in B_c} l(w,x,y)$ to denote the empirical loss over the stored data of client c.

In this work, we investigate the impacts of each client's limited storage on FL, and consider the widely adopted algorithm Fed-Avg [9] for easy illustration. Under the synchronous FL framework, the global model is trained by repeating the following two steps for each communication round t:

[1]We will use mobile devices and clients interchangeably in this work.

(i) Local Training: In round t, the server selects a client subset $C_t \subseteq C$ to participate in model training process. Each participating client $c \in C_t$ downloads the current global model w_{fed}^{t-1} (the ending global model of the last round), and performs m epochs of model updates with the locally stored data:

$$w_c^{t,i} \leftarrow w_c^{t,i-1} - \eta \nabla_w \tilde{F}_c(w_c^{t,i-1}),\ i = 1, \cdots, m, \quad (1)$$

where the starting model $w_c^{t,0}$ is initialized as w_{fed}^{t-1}, and η denotes the learning rate.

(ii) Model Aggregation: Each participating client $c \in C_t$ uploads the updated local model $w_c^{t,m}$, and the server aggregates them to derive a new global model w_{fed}^t:

$$w_{\text{fed}}^t \leftarrow \sum_{c\in C_t} \zeta_c^t \cdot w_c^{t,m}, \quad (2)$$

where $\zeta_c^t = \frac{v_c}{\sum_{c'\in C_t} v_{c'}}$ is the normalized weight of the participating client c in current round t.

In the scenario of FL with limited on-device storage and streaming networked data, we have an additional data selection step for each client:

(iii) Data Selection: In each round t, once receiving a new data sample, the client has to make an online decision on whether to store the new sample (in place of an old one if the storage area is fully occupied) or discard it. The goal of this process is to select valuable data samples from on-device streaming data for model training in the upcoming rounds.

III. ODE Design

In this section, we first quantify the impact of a local data sample on the performance of global FL model in terms of convergence rate and inference accuracy. Based on the common dominant term in these two analytical expressions, we propose a new data valuation metric for data evaluation and selection in FL (Section III-A), and develop a practical method to estimate this metric with low computation and communication overhead (Section III-B). We further design a strategy for the server to coordinate cross-client data selection process, avoiding the potential overlapped data selected and stored by different clients (Section III-C). Finally, we summarize the procedure of ODE (Section III-D).

A. Data Valuation Metric

We evaluate the impact of a local data sample on FL model from the perspectives of convergence rate and inference accuracy, which are two critical aspects for FL. The convergence rate quantifies the reduction of loss function in each training round, and determines the communication cost of FL. The inference accuracy reflects the performance of a model on guaranteeing the quality of application service and user experience. For theoretical analysis, we follow one typical assumption, which is widely adopted in previous literature.

Assumption 1 (Lipschitz Gradient): For each client $c \in C$, the loss function $F_c(w)$ is L_c-Lipschitz gradient, i.e., $\| \nabla_w F_c(w_1) - \nabla_w F_c(w_2) \|_2 \leqslant L_c \| w_1 - w_2 \|_2$, *which implies that the global loss function $F(w)$ is L-Lipschitz gradient with $L = \sum_{c\in C} \zeta_c L_c$.*

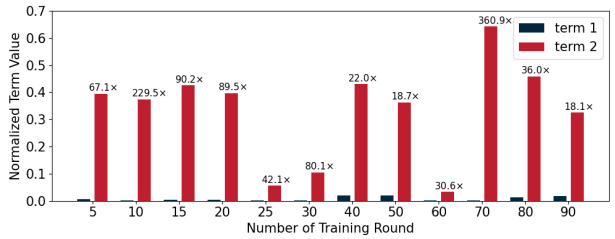


Fig. 2. Normalized mean values of term 1 and term 2 across $35,000+$ data samples in HAR task across different training rounds.

Convergence Rate. We provide a lower bound on the reduction of loss function of global model after model aggregation in each communication round of FL.

Theorem 1 (Global Loss Reduction): With Assumption 1, for an arbitrary set of clients $C_t \subseteq C$ selected by the server in round t, the reduction of global loss $F(w)$ is bounded by:

$$\underbrace{F(w_{\text{fed}}^{t-1}) - F(w_{\text{fed}}^{t})}_{\text{global loss reduction}} \geqslant \sum_{c\in C_t}\sum_{i=0}^{m-1}\sum_{(x,y)\in B_c}\Big[-\alpha_c \underbrace{\| \nabla_w l(w_c^{t,i},x,y)\|_2^2}_{\text{term 1}} + \beta_c \underbrace{\langle \nabla_w l(w_c^{t,i},x,y), \nabla_w F(w_{\text{fed}}^{t-1})\rangle}_{\text{term 2}}\Big], \tag{3}$$

where $\alpha_c = \frac{L\zeta_c^t}{2}\cdot\left(\frac{\eta}{|B_c|}\right)^2$ *and* $\beta_c = \zeta_c^t\cdot\left(\frac{\eta}{|B_c|}\right)$.

Proof: Please refer to supplementary material. □

Theorem 1 demonstrates that the loss reduction of the global FL model w_{fed}^t in each round is closely related to two terms: gradient magnitude of the local data sample (term 1) and the projection of the local gradient of a data sample onto the global gradient over global data distribution (term 2). We demonstrate that term 2 has a significantly greater influence than term 1 from two perspectives. First, the coefficient ratio between term 2 and term 1 is approximately $\frac{\beta_c}{\alpha_c}\approx 40^4$ in typical FL scenarios with learning rate 0.001 and on-device storage capacity larger than 10 samples. Additionally, our experimental results (Figure 2) on a real-world human activity recognition task of over $35,000$ data samples illustrate that: on average, the value of term 2 surpasses the value of term 1 by a factor of $90\times$ across various global models in different training rounds. Consequently, we can focus on term 2 (*i.e.*, projection of the local gradient of a data sample onto the global gradient) when evaluating the impact of a local data sample on the convergence rate of global model.

We next briefly describe how to evaluate data samples based on term 2. The local model parameter $w_c^{t,i}$ in term 2 is computed from Eq. (1), where the gradient $\nabla_w \tilde{F}_c(w_c^{t,i-1})$ depends on all the locally stored data samples. In other words, the value of term 2 depends on the "cooperation" of all the locally stored data samples. Thus, we can formulate the computation of term 2 via cooperative game theory, where each data sample represents a player and the utility of each sample set is the value of term 2. Within this cooperative game, the individual contribution of each data sample to term 2 can be regarded as its value, quantified through leave-one-out [15] or Shapley Value [22]. However, these methods necessitate multiple times of model retraining to compute the marginal contribution of each data sample, and thus we propose a one-step look-ahead policy to estimate each sample's value by only focusing on the first local training epoch ($m=1$).

Inference Accuracy. We can assume that the optimal model is obtained by gathering all clients' generated data and conducting CL. Moreover, as the testing dataset and the corresponding testing accuracy are hard to obtain in FL, we propose to leverage the weight divergence between the models trained through FL and CL, *i.e.*, $\| w_{\text{fed}}^t - w_{\text{cen}}^{mt}\|$, to quantify the accuracy of the FL model in each round t. With $t\to\infty$, we can evaluate the final accuracy of FL model.

Theorem 2 (Model Weight Divergence): With Assumption 1, for an arbitrary participating client set C_t, we have the following inequality for the weight divergence between the models trained through FL and CL after the t^{th} training round.

$$\begin{aligned}&\| w_{\text{fed}}^t - w_{\text{cen}}^{mt}\|_2 \\ &\leqslant (1+\eta L)^m \| w_{\text{fed}}^{t-1} - w_{\text{cen}}^{m(t-1)}\|_2 \\ &\quad + \sum_{c\in C_t}\zeta_c^t\left[\eta\sum_{i=0}^{m-1}(1+\eta L)^{m-1-i} G_c(w_c^{t,i})\right],\end{aligned} \tag{4}$$

where $G_c(w) = \| \nabla_w \tilde{F}_c(w) - \nabla_w F(w)\|_2$.

Proof: Please refer to supplementary material. □

The following lemma further shows the impact of a local data sample on weight divergence $\| w_{\text{fed}}^t - w_{\text{cen}}^{mt}\|_2$ through $G_c(w_c^{t,i})$.

Lemma 1 (Gradient Divergence): For an arbitrary client $c\in C$, $G_c(w) = \| \nabla \tilde{F}_c(w) - \nabla F(w)\|_2$ is bounded by:

$$G_c(w) \leqslant \left[\underbrace{\| \nabla_w F(w)\|_2^2}_{\text{constant}} + \sum_{(x,y)\in B_c}\frac{1}{|B_c|}\Big(\underbrace{\| \nabla_w l(w,x,y)\|_2^2}_{\text{term 1}} - 2\underbrace{\langle \nabla_w l(w,x,y), \nabla_w F(w)\rangle}_{\text{term 2}}\Big)\right]^{1/2}. \tag{5}$$

Proof: Please refer to supplementary material. □

Due to distinct coefficients, the twofold gradient projection (term 2) naturally exhibits a larger mean and variance compared with the gradient L2-norm (term 1) across various data samples. Also, as mentioned before, our experimental results shown in Figure 2 demonstrate the contrasting magnitude orders of term 1 and term 2 (*e.g.*, value of term 2 surpasses value of term 1 by a factor of $90\times$ on the real-world HAR task). As a result, we can quantify the impact of a local data sample on $G_c(w)$ and the inference accuracy mainly through term 2, which happens to be the same as the term 2 in (3).

Data Valuation. Based on the above analysis, we define a new data valuation metric for FL, and provide theoretical understanding as well as intuitive interpretation.

Definition 1 (Data Valuation Metric): In the t^{th} round, for a client $c\in C$, the value of a data sample (x,y) is defined as the projection of its local gradient $\nabla l(w,x,y)$ onto the gradient of current global model over the global data distribution:

$$v(x,y) \overset{\text{def}}{=} \langle \nabla_w l(w_{\text{fed}}^t, x, y), \nabla_w F(w_{\text{fed}}^t)\rangle. \tag{6}$$

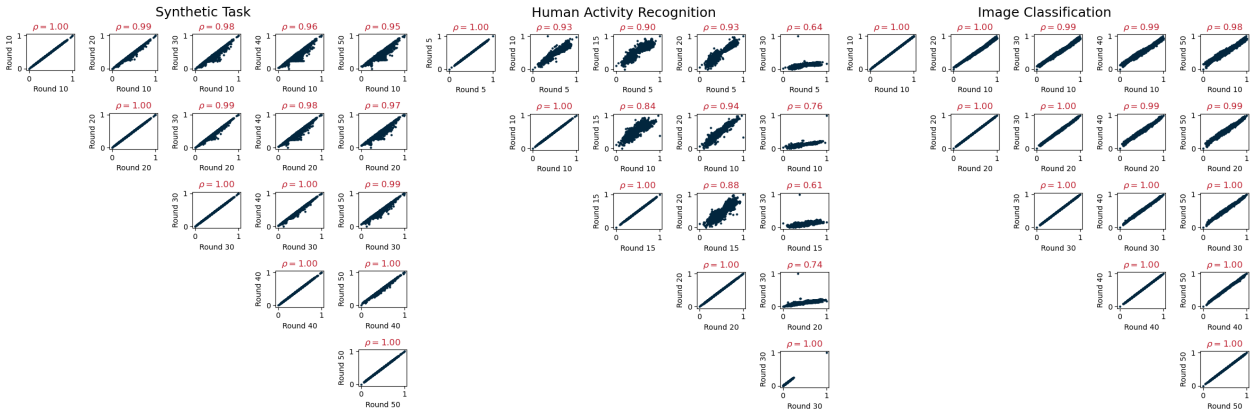


Fig. 3. Experimental results on three FL tasks to support that the data values remain stable across a few training rounds.

Based on this new data valuation metric, once a client receives a new data sample, she can make an online decision on whether to store this sample by comparing its data value with values of old samples in storage, which can be easily implemented as a priority queue on each client.

Theoretical Understanding. On one hand, maximizing the above data valuation metric of the selected data samples is a one-step greedy strategy for minimizing the loss of the updated global model in each training round, because it optimizes the dominant term 2 in the lower bound of global loss reduction in Eq. (3). The one-step-look-ahead policy means that we only consider the first epoch of the local model training in Eq. (3), *i.e.*, $\langle \nabla_w l(w_c^{t,i}, x, y), \nabla_w F(w_{\text{fed}}^{t-1}) \rangle$ with index i set as 0. On the other hand, this metric also improves the inference accuracy of the final global model by narrowing the gap between the models trained through FL and CL, as it reduces the high-weight term of the dominant part in Eq. (4), *i.e.*, $(1+\eta L)^{m-1} G_c(w_k^{t,0})$.

Intuitive Interpretation. Large value of a data sample indicates that its impact on the global FL model is similar to that of the underlying global data distribution, guiding the clients to select the data samples which not only follow their own local data distribution but also have similar effects with the global data distribution. In this way, part of the personalized information of each client's local data is preserved while the data heterogeneity across clients is also reduced, which have been demonstrated to improve FL performance [23], [24], [25].

B. On-Client Data Selection

In practice, it is non-trivial for one client to directly utilize the above data valuation metric for online data selection due to the following two challenges: (i) Lack of the latest global model: due to the partial participation of clients in FL, each client $c \in C$ cannot derive the global FL model w_{fed}^{t-1} in the rounds that she is not selected for model training, and only has the outdated global FL model from the previous participating round, *i.e.*, $w_{\text{fed}}^{t_{c,\text{last}}-1}$. (ii) Lack of unbiased global gradient: the accurate global gradient over the unbiased global data distribution can only be obtained by aggregating all the clients' local gradients over their unbiased local data distributions. This is hard to achieve because only partial clients participate in each communication round, and the locally stored data could become biased during the on-client data selection process.

We can consider that challenge (i) does not affect the online data selection process too much and clients can simply leverage the outdated global model for data valuation. This relies on our experimental observation in Figure 3 that the value of each data sample remains stable across a few training rounds. For example, the Pearson correlation coefficients between data values computed with FL models in rounds t and $t+10$ consistently surpass 0.85 for all the three typical FL tasks.

To overcome challenge (ii), we propose a gradient estimation method. First, to solve the issue of skew local gradient, each client $c \in C$ is allowed to maintain a local gradient estimator $\hat{g}_c$, which is continuously updated whenever the n^{th} streaming data sample (x, y) is generated by the client:

$$\hat{g}_c \leftarrow \frac{n-1}{n}\hat{g}_c + \frac{1}{n}\nabla_w l(w_{\text{fed}}^{t_{c,\text{last}}-1}, x, y), \tag{7}$$

where $t_{c,last}$ denotes the last participating round of client c, and thus $w_{\text{fed}}^{t_{c,last}-1}$ denotes the latest global model that client c received. Essentially, the local gradient estimator works as a running sum estimation of the gradient of the underlying local data distribution, which approximates the average gradient of all data samples generated by client c since last participating round. When the client c is selected to participate in FL in a certain round t, the client uploads the current local gradient estimator $\hat{g}_c$ to server, and resets the local gradient estimator as a new global FL model w_{fed}^{t-1} is received, *i.e.*, $\hat{g}_c \leftarrow 0, n \leftarrow 0$. Second, to solve the problem of skew global gradient caused by partial client participation, the server is requested to maintain a global gradient estimator $\hat{g}^t$, which is an aggregation of the local gradient estimators, i.e. $\hat{g}^t = \sum_{c \in C} \zeta_c \hat{g}_c$. As it incurs high communication cost to collect $\hat{g}_c$ from all clients, the server only uses $\hat{g}_c$ of the participating clients to update global gradient estimator $\hat{g}^t$ in each round t: $\hat{g}^t \leftarrow \hat{g}^{t-1} + \sum_{c \in C_t} \zeta_c (\hat{g}_c - \hat{g}_c^{t_{\text{last}}})$. The rationale behind the global gradient estimator lies in the high similarity between the gradients of each client's local data distribution over global models in two consecutive participating rounds, as demonstrated empirically in Figure 3. Thus, in each round

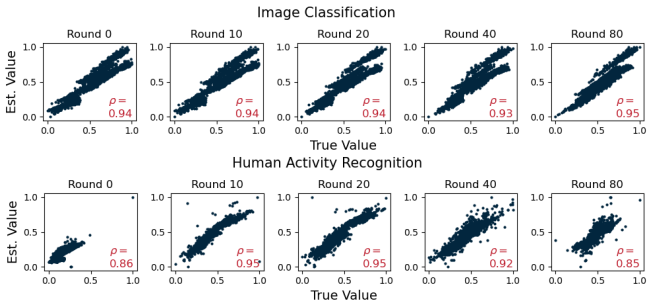


Fig. 4. Experiments results to demonstrate the high similarity between data values computed from partial model and full model.

t, the server needs to distribute both the current global model w_{fed}^{t-1} and the latest global gradient estimator $\hat{g}^{t-1}$ to each selected client $c \in C_t$, who will conduct local model training and upload both locally updated model $w_c^{t,m}$ and local gradient estimator $\hat{g}_c$ back to the server.

Simplified Version. In both of the local gradient estimation in Eq. (7) and data valuation in Eq. (6), for a new data sample, we need to backpropagate the entire ML model to compute the gradient, which introduces high computation cost and memory footprint for buffering the intermediate outputs of model. Consequently, we propose to use the gradients of the last few network layers of ML models instead of the whole model for data selection. The success of such simplification relies on our experimental observation in Figure 4 that partial model gradient is able to reflect the trend of the full model gradient. Specifically, the Pearson correlation coefficient between data values computed from only last model layer and full model layers consistently exceeds 0.93 for image classification task and 0.85 for human activity recognition task.

Privacy Concern. Uploading local gradient estimators may disclose the private information of each client, which can be mitigated through differential privacy [26]. Specifically, differential privacy empowers clients to add a controlled amount of Gaussian noise to local gradient estimator for obfuscating the true information, while ensuring that the aggregated global gradient estimator remains unbiased and meaningful.

C. Cross-Client Data Storage

Since the local data distributions of different clients may overlap with each other, independently conducting data selection process for each client may lead to the distorted distribution of data stored by all clients.

One potential solution is to divide the global data distribution into several regions, and coordinate each client to store valuable data samples for one specific distribution region, while the union of all stored data still follow the unbiased global data distribution. In this work, we consider the label of data samples as the dividing criterion.[2] Thus, before the training process, the server needs to instruct each client the labels and the corresponding quantity of data samples to store. Considering the partial client participation and heterogeneous data distribution among clients, the cross-client coordination policy need to satisfy the following four desirable properties:
(i) Efficient Data Selection: To improve the efficiency of data selection, the label $y \in Y$ should be assigned to the clients who generate more data samples with this label, following the intuition that there is a higher probability to select more valuable data samples from a larger candidate dataset.
(ii) Redundant Label Assignment: To ensure that all the labels are likely to be covered in each round even with partial client participation, each label $y \in Y$ is required to be assigned to more than n_y^{label} clients, which is a hyperparameter decided by the server.
(iii) Limited Storage: Due to limited on-device storage, each client c should be assigned to less than n_c^{client} labels to ensure a sufficient number of valuable data samples stored for each label, and n_c^{client} is a hyperparameter decided by the server;
(iv) Unbiased Global Distribution: The weighted average of all clients' stored data distribution is expected to be equal to the unbiased global data distribution, *i.e.*, $\tilde{P}(y) = P(y), y \in Y$.

Problem Formulation. We represent the cross-client data storage strategy as a coordination matrix $D \in \mathbb{N}^{|C|\times|Y|}$, where $D_{c,y}$ denotes the number of data samples with label y that client c is allowed to store. We use matrix $V \in \mathbb{R}^{|C|\times|Y|}$ to denote the statistical information of each client's generated data, where $V_{c,y} = v_c P_c(y)$ denotes the average speed of the data samples with label y generated by client c. The cross-client coordination algorithm can be obtained by solving an optimization problem, where the four properties are mathematically expressed as the objective and three constraints. Specifically, the objective is to maximize the dot projection between coordination matrix D and information matrix V to optimize the data selection efficiency of property (i). Remaining properties (ii)-(iv) can be formulated as constraints on each column and row of the coordination matrix D:

$$\max_{D} \quad \sum_{c\in C}\sum_{y\in Y} D_{c,y}\cdot V_{c,y} = \| DV \|_1, \tag{8}$$

$$\text{s.t.} \quad \| D_y^{\mathrm{T}} \|_0 \geqslant n_y^{\text{label}}, \quad \forall y \in Y, \tag{9}$$

$$\| D_c \|_0 \leqslant n_c^{\text{client}}, \quad \forall c \in C, \tag{10}$$

$$\| D_c \|_1 = |B_c|, \quad \forall c \in C,$$

$$\frac{\sum_{c\in C} D_{c,y}}{\| B \|_1} = \frac{\sum_{c\in C} V_{c,y}}{\| V \|_1}, \quad \forall y \in Y. \tag{11}$$

Complexity Analysis. We can verify that the above optimization problem with l_0 norm is a convex-cardinality problem, which is NP-hard [27]. The globally optimal solution can be obtained through the following two steps: (i) Decide the feasible region: determine the non-zero elements within the coordination matrix D, *i.e.* $S = \{(c,y)|D_{c,y} \neq 0\}$, to satisfy constraints (9) and (10); (ii) Solve the simplified problem: determine the specific values of the non-zero elements in S by solving a simplified optimization problem:

$$\max_{D} \quad \sum_{c\in C, y\in Y, (c,y)\in S} D_{c,y}\cdot V_{c,y}$$

$$\text{s.t.} \quad \sum_{y\in Y,(c,y)\in S} D_{c,y} = |B_c|, \quad \forall c \in C,$$

[2]There are some other methods to divide the data distribution, such as K-means and hierarchical clustering, and our results are independent on them.

$$\frac{\sum_{c\in C,(c,y)\in S} D_{c,y}}{\parallel D \parallel_1} = \frac{\sum_{c\in C} V_{c,y}}{\parallel V \parallel_1}, \quad \forall y \in Y. \quad (12)$$

Compared with the original optimization problem, the objective (8) remains unchanged and the constraints (9) and (10) are eliminated. Consequently, the solution to the simplified problem (12) can be regarded as part of the solution to the original optimization problem (8)-(11). As the number of possible S can be exponential to $|C|$ and $|Y|$, it is impractical to solve the simplified problem for exponential times to derive the globally optimal solution D. The classic approach is to replace all the non-convex discontinuous l_0 norm constraints with the convex and continuous l_1 norm [27], which fails to work in our scenario because simultaneously minimizing as many as $|C|+|Y|$ l_1 norms in the objective function will lead to high computation and memory costs as well as unstable solutions [28]. As a result, we propose an intuitive and greedy algorithm that enables the direct identification of the non-zero elements S in the coordination matrix D. The key insight of our greedy algorithm lies in identifying S with the highest potential value of objective function (8).

Algorithm 1 Greedy Cross-Client Coordination

Input: Label vector Y, client vector C, data velocity matrix V, storage size matrix B, redundant label assignment n^{label}, limited storage n^{client}
Output: Coordination matrix D

```
// Sort labels by number of owners
Y ← Sort Y in an increasing order of the number of non-zero elements in V[:, y];
for y in Y do
    // Sort clients by data velocity
    C_tmp ← Sort c ∈ C in an decreasing order of the data velocity V[c, y];
    for c in C_tmp do
        // Limited storage requirement
        if Sum(D[c, :]) < n_c^client then
            D[c, y] ← 1;
        end
        // Redundant label requirement
        if Sum(D[:, y]) ≥ n_y^label then
            Break;
        end
    end
end
// Divide each device storage evenly
for c in C do
    cnt ← Sum(D[c, :]);
    for y in Y do
        D[c, y] ← B[c, y] * D[c, y]/cnt;
    end
end
return D;
```

Greedy Cross-Client Coordination. We summarize the greedy algorithm in Algorithm 1 and provide an illustrative example in Figure 5, which consists of three steps:
(i) Information Collection: Each client $c\in C$ sends the rough information about its local data to the server, including the storage capacity $|B_c|$ and data velocity $V_{c,y}$ for each label y, which can be obtained from the statistics of previous time periods. Based on this information, the server constructs the storage size vector $B \in \mathbb{N}^{|C|}$ and the data velocity matrix $V \in \mathbb{R}^{|C|\times|Y|}$ for all clients. For example, in Figure 5, clients c_1, c_2, c_3 upload server the storage space of $|B_1| = 20, |B_2| = 15, |B_3| = 15$ and data velocity vectors of $V_1 = [400, 200, 300], V_2 = [200, 300, 200], V_3 = [300, 0, 400]$.
(ii) Label Assignment: Aligned with the objective (8), our goal is to assign each label to clients with higher generation velocities for that label. This can be achieved by ranking clients based on their corresponding velocities for each label and allocating each label to the top few clients within the constraint (9). However, constraint (10) limits the number of data labels each client can store, which may contradict with the allocation results computed previously. We note that fulfilling constraint (9) is more challenging than constraint (10) due to the existence of scarce labels with few owners, and thus ODE prioritizes the allocation of such scarce labels. For implementation, the server first sorts labels Y according to the number of owners (line 1-2). Then, for each considered label $y \in Y$ in the order, the server sorts all clients based on their generation velocities (line 3-5), and then the label y is allocated to the top clients within constraint (10), which can be achieved by removing a client c from the sorting order once the number of its assigned labels exceeds n_c^{client} (line 6-10). In Figure 5, central server first sorts the clients for each label based on the clients' generation velocities to obtain the sorting order (c_1, c_3, c_2) for label y_1, (c_2, c_1) for label y_2, (c_3, c_1, c_2) for label y_3. Next, the server sorts the data labels in a non-decreasing order based on the number of owners and obtains the label order (y_2, y_1, y_3). Based on the above sorting orders and constraints (9) and (10), server assigns label y_2 to clients (c_1, c_2), label y_1 to clients (c_1, c_3), and label y_3 to clients (c_3, c_2) for data selection and storage.
(iii) Quantity Assignment: With the two steps mentioned above, we have determined the non-zero elements of the client-label coordination matrix D. To further reduce computational complexity and avoid imbalanced on-device data storage for each label, we do not directly solve the simplified optimization problem (12), and require each client to evenly divide the storage capacity among the assigned labels. To ensure that the weighted distribution of the stored data approximates the unbiased global data distribution, we compute a weight γ_y for each label $y\in Y$ to satisfy $\gamma_y \tilde{P}(y) = P(y)$ (line 18-23):

$$\gamma_y = \frac{P(y)}{\tilde{P}(y)} = \frac{\parallel V^{\mathrm{T}}[y] \parallel_1 / \parallel V \parallel_1}{\parallel D^{\mathrm{T}}[y] \parallel_1 / \parallel D \parallel_1}. \quad (13)$$

As a result, each client only needs to maintain one priority queue with a size of $\frac{|B_c|}{\|D_c\|_0}$ for each assigned label. During local model training, each participating client $c\in C_t$ updates its local model using the weighted stored data samples:

$$w_c^{t,i} \leftarrow w_c^{t,i-1} - \frac{\eta}{\zeta_c} \sum_{(x,y)\in B_c} \gamma_y \nabla_w l(w_c^{t,i-1}, x, y), \quad (14)$$

where $\zeta_c = \sum_{(x,y)\in B_c} \gamma_y$ denotes the new weight of each client, and the normalized weight of client $c\in C_t$ for model aggregation in round t becomes $\zeta_c^t = \sum_{c\in C_t} \frac{\zeta_c}{\sum_{c'\in C_t} \zeta_{c'}}$. Take client c_1 in Figure 5 for example, for label y_1, its fraction in the whole storage space is $\frac{10+0+10}{20+40+20} = \frac{1}{4}$, while its fraction in all the clients' generated data samples is $\frac{400+200+300}{900+700+700} = \frac{9}{23}$. Therefore, the weight of label y_1 on client c_1 is $\frac{9/23}{1/4} = \frac{36}{23}$.

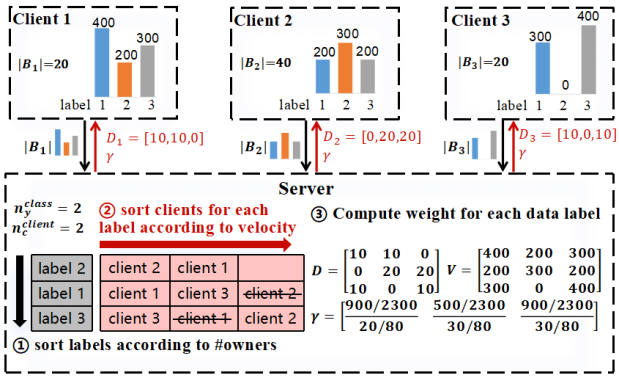


Fig. 5. A simple example to illustrate the greedy algorithm.

Approximation Ratio. Given that it is difficult to theoretically analyze the approximation ratio of the greedy Algorithm 1, we empirically demonstrate its effectiveness in estimating the globally optimal solution to the original optimization problem (8)-(11) under various FL settings. For simplicity, we set the class number to 10 and the hyper-parameters to $n_c^{client}=2$ and $n_y^{label}=2$. In each experiment, we sample each client's data velocity matrix V_c from a uniform distribution, and compare the objective function values of solutions obtained by four methods: Optimal (traversing the entire feasible region to find the globally optimal solution), Greedy (Algorithm 1), Random (average result of 20 solutions sampled randomly from the feasible region), and Worst (worst result of 20 random solutions). Empirical results shown in Figure 6 illustrate that our greedy algorithm consistently achieves an approximation ratio exceeding 0.8 across various FL settings.

Privacy Concern. The potential leakage of such rough information is generally acceptable to clients, since it does not expose the specific raw data samples. To further address the privacy concern, fully homomorphic encryption sorting can be employed to sort k encrypted elements with computational complexity $O(\log^2 k)$ [29]. Consequently, the central server can leverage the encrypted information of each client to obtain cross-client coordination strategy by (i) sorting $|Y|$ data classes with complexity $O(\log^2 |Y|)$, and (ii) sorting the clients for each class with total complexity $O(|Y| \log^2 |C|)$.

D. Overall Procedure of ODE

As illustrated in Figure 7, ODE incorporates cross-client data storage and on-client data evaluation to coordinate clients to collaboratively store valuable samples for FL, simultaneously speeding up model training process and improving inference accuracy of the final global model.

Key Idea. The cross-client data storage component coordinates clients to store the valuable data samples with different labels to avoid highly overlapped data stored by all clients. The on-client data evaluation component instructs each client to select the data having similar local gradient with the global gradient, which reduces data heterogeneity among clients while also preserves part of the personal information as the data samples are selected from the true local data distribution.

Cross-Client Data Storage. Before the FL model training process, the central server collects data distribution information and storage capacity from all clients (①), and greedily solves the optimization problem in Eq. (8)-(11) (②).

On-Client Data Evaluation. During the FL process, each client trains the local model in participating rounds, and leverages idle computation and memory resources to perform on-device data selection in non-participating rounds, which introduces little extra time overhead to each FL training round. In the t^{th} training round, non-selected clients, selected clients and server execute different processes:

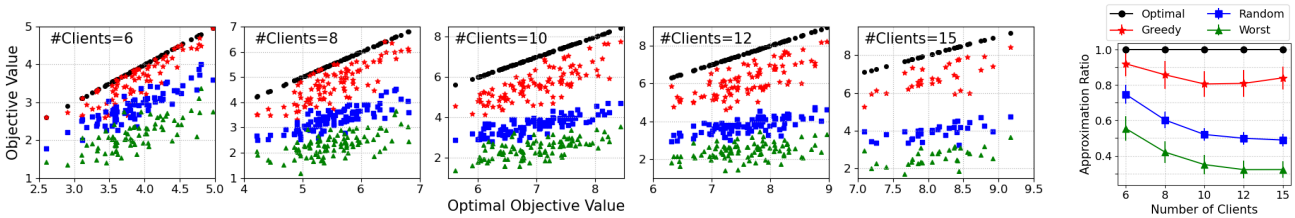
- **Non-Selected Clients.** Data evaluation (③): each non-selected client $c \in C \setminus C_t$ steadily evaluates and selects the streaming data samples according to the data valuation metric in Eq. (6), where the clients leverage the global gradient estimator received in last participating round instead of the accurate one for data evaluation. Local gradient estimator update (④): the client also dynamically updates the local gradient estimator $\hat{g}_c$ with real-time data through Eq. (7).
- **Selected Clients.** Local model update (⑤): after receiving the new global model w_{fed}^{t-1} and new global gradient estimator $\hat{g}^{t-1}$ from the server, each selected client $c \in C_t$ performs local model updates using the stored data samples by Eq. (14). Local model and estimator transmission (⑥): each selected client uploads the updated model $w_c^{t,m}$ and local gradient estimator $\hat{g}_c$ to server. Then, the local estimator is reset to approximate the local gradient of the newly received global model.
- **Server.** Global model and estimator transmission (⑦): at the beginning of each round t, the server distributes the global model w_{fed}^{t-1} and the global gradient estimator $\hat{g}^{t-1}$ to the selected clients C_t. Local model and estimator aggregation (⑧): at the end of each round, the server collects the updated local models $w_c^{t,m}$ and local gradient estimators $\hat{g}_c$ from participating clients $c \in C_t$ to derive the new global model and update the global gradient estimator $\hat{g}^t$.

IV. Evaluation

In this section, we first introduce experiment setting, baselines and evaluation metrics (Section IV-A). Second, we show the individual and integrated impacts of the limited on-device storage and streaming networked data on FL to demonstrate the motivation of our work (Section IV-B). Third, we present the overall performance of ODE and baselines, including model training speedup, final inference accuracy, memory footprint and data evaluation delay (Section IV-C). Next, we test the robustness of ODE against various environmental factors (Section IV-D), such as the number of local training epochs m, client participation rate $\frac{|C_t|}{|C|}$, on-device storage capacity B_c, mini-batch size, data heterogeneity across clients and varied levels of statistical error in clients' velocities. Finally, we analyze the component-wise effect of ODE (Section IV-E).

A. Experiment Setting

Learning Tasks, Datasets and ML Models. To demonstrate the ODE's superior performance and generalization across various learning tasks, datasets and ML models. We evaluate ODE on one synthetic dataset, two real-world



(a) Approximation performance on cases with different numbers of clients. (b) Approximation ratio.

Fig. 6. Comparison of the objective function values obtained by four different methods: Optimal, Greedy, Random and Worst Case.

TABLE I
STATISTICS OF DIFFERENT TASKS, DATASETS, MODELS AND DEFAULT EXPERIMENT SETTINGS

Tasks	Datasets	Models	#Samples	#Labels	#Devices	n_y^{label}	$\frac{\lvert C_t\rvert}{\lvert C\rvert}$	η	m	$\lvert B_c\rvert$
ST	Synthetic Dataset [18]	LogReg	$1,016,442$	10	200	5	5%	$1e^{-4}$	5	10
IC	Fashion-MNIST [19]	LeNet [30]	$70,000$	10	50	5	10%	$1e^{-3}$	5	5
HAR	HARBOX [20]	Customized DNN	$34,115$	5	120	5	10%	$1e^{-3}$	5	5
TC	Industrial Dataset	Customized CNN	$37,707$	20	30	5	20%	$5e^{-3}$	5	10

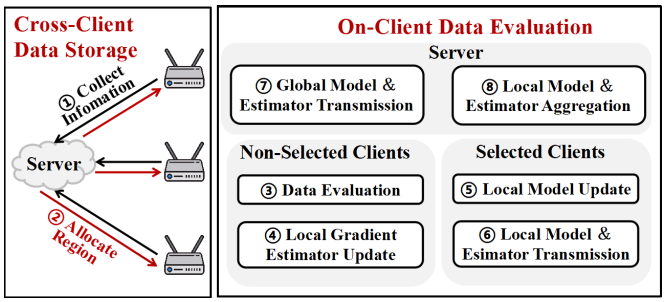


Fig. 7. Overview of ODE framework.

datasets and one industrial dataset, all of which vary in data quantities, distributions and model outputs, and cover the tasks of synthetic task (ST), image classification (IC), human activity recognition (HAR) and network traffic classification (TC). The statistics of these tasks are summarized in Table I:

(i) Synthetic Task. The synthetic dataset we used is proposed in LEAF benchmark [18], which contains 200 clients and 1 million data samples, and a logistic regression model is trained for this 10-class classification task.

(ii) Image Classification. Fashion-MNIST [19] contains $60,000$ training images and $10,000$ testing images, which are divided into 50 clients according to labels [9]. We train LeNet [30] for this 10-class image classification task.

(iii) Human Activity Recognition. HARBOX [20] is a 9-axis IMU dataset collected from 121 users' smartphones in a crowdsourcing manner, including 34,115 data samples with 900 dimension. Considering the simplicity of the dataset, a lightweight customized DNN with two dense layers followed by a $SoftMax$ layer is deployed for this 5-class human activity recognition task.

(iv) Network Traffic Classification. The industrial dataset of mobile application classification is collected from 30 ONT devices in a simulated network environment from May 2019 to June 2019. Generally, the dataset contains more than $560,000$ data samples and 250 mobile applications as labels, which cover the categories of video, game, file downloading and communication. We manually label the application of each data sample. The model we leverage is a typical CNN consisting of 4 convolutional layers with kernel size 1×3 to extract features and 2 fully-connected layers for classification. To reduce the training time caused by the large scale of dataset, we uniformly select 20 out of 250 applications with a total number of $37,707$ data samples as training data.

Parameters Configurations. For all the experiments, we use SGD as the optimizer and decay the learning rate per 100 rounds by $\eta_{\text{new}} = 0.95 \times \eta_{\text{old}}$. To simulate the setting of streaming networked data, we set the on-device data velocity to be $v_c = \frac{\#\text{training samples}}{500}$, which means that each device $c \in C$ will receive v_c data samples one by one in each communication round, and the samples would be shuffled and appear again per 500 rounds. Other default configurations are shown in Table I. Note that the participating clients in each round are randomly selected, and for each experiment, we repeat 5 times and present the average results.

Baselines. In our experiments, we compare two versions of ODE, *ODE-Exact* (using exact global gradient) and *ODE-Est* (using estimated global gradient), with four categories of data selection baselines, including random sampling methods (*RS*), importance-based methods for CL (*HL* and *GN*), existing data selection methods for FL (*FB* and *SLD*) and the ideal case with unlimited on-device storage (*FD*):

(i) *Random sampling (RS)* includes Reservoir Sampling [21] and First-In-First-Out (storing the latest $|B_c|$ data samples). As their experimental results are comparable, we show the results of First-In-First-Out for *RS* only.

(ii) Importance sampling-based methods include *HighLoss (HL)*, which uses the loss of each data sample as value for selection [31], [32], and *GradientNorm (GN)*, which quantifies the data value through gradient norm [17], [33].

(iii) Existing data selection methods for FL includes *FedBalancer (FB)* [13] and *SLD* (Sample-Level Data selection) [14], which are slightly revised for adapting to streaming data. Specifically, we store the loss/gradient norm of the latest 50 samples for noise removal. For *FB*, we ignore the data samples with top 10% loss values, and for *SLD*, we remove

TABLE II
IMPACT OF LIMITED ON-DEVICE STORAGE ON TRAFFIC CLASSIFICATION TASK WITH DIFFERENT SETTINGS, WHERE THE CONVERGENCE TIME IS NORMALIZED BY THE TIME OF *FD*

Settings	#Label per Client	Average Variance	#Local Epoch	Converg-ence Time	Final Accuracy	
					RS	FullData
Setting 1	5	0.227	2	2.18	0.867	0.912
			5	1.70	0.855	0.890
Setting 2	10	0.234	2	2.50	0.894	0.932
			5	2.84	0.874	0.926
Setting 3	15	0.235	2	3.46	0.906	0.960
			5	3.92	0.890	0.957

the samples with gradient norm larger than the median norm value.

(iv) Ideal method with unlimited on-device storage, denoted as *FullData (FD)*, using the entire dataset of each client for training to simulate the unlimited storage scenario.

Metrics for Training Performance. We use two metrics to evaluate the performance of each data selection method: (i) *Time-to-Accuracy Ratio*: we measure the model training speedup by the ratio of training time of *RS* and the considered method to reach the same target accuracy, which is set to be the final inference accuracy of *RS*. As the time of one communication round is usually fixed in practical FL scenario, we can quantify the training time with the number of communicating rounds. (ii) *Final Inference Accuracy*: we evaluate the inference accuracy of the final global model on each device's testing data, and report the average accuracy for evaluation.

B. Experiment Results for Motivation

In this subsection, we provide the comprehensive experimental results for the motivation of our work. First, we demonstrate the severe impact of limited on-device storage and streaming networked data on classic FL process in various settings, such as different numbers of local training epochs and various clients' data. Then, we analyze the separate impacts of theses two properties.

Overall Impact. To investigate the impact of limited on-device storage and steaming networked data on FL with different data settings, such as the range and variance of local data distribution, we conduct experiments over three settings with different numbers of labels owned by each client, which is constructed from the industrial TC dataset. The experimental results shown in Table II demonstrate that: (i) With local data variance increasing, the model training speed and final accuracy drops significantly, as the stored data is more likely to be biased when the underlying data distibution has a wide range; (ii) When the number of local training epochs increases, the negative impact of two properties becomes more serious due to large model updates toward the biased direction.

Individual Impact. To show the individual impact of streaming networked data, we ignore limited storage by assuming that each device can utilize all the generated data instead of only stored data for online data evaluation and selection. To show the individual impact of limited storage, we ignore streaming networked data by supposing that in each epoch, all the new data samples are generated concurrently and can be accessed arbitrarily. The experimental results shown in Table III demonstrates that limited storage is the root cause of the performance degeneration of FL process, because (i) streaming data mainly prevents previous methods from making accurate online decisions on data selection, as they select each sample according to a normalized probability depending on both discarded and upcoming samples, which are not available in streaming setting but can be estimated; (ii) the limited storage prevents previous methods from deriving complete information of local and global data and guides clients to select suboptimal data from an insufficient candidate dataset.

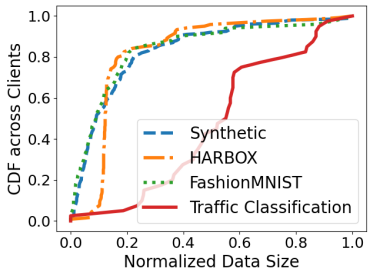


Fig. 8. Unbalanced data sizes of clients.

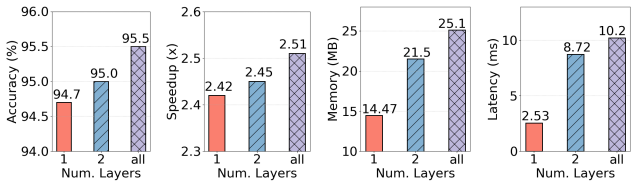


Fig. 9. Performance of ODE with different numbers of model layers for data evaluation and selection on industrial TC task.

TABLE III
IMPACTS OF LIMITED STORAGE AND STREAMING DATA ON FL WITH ST DATASET. SYMBOL '–' MEANS FAILING TO REACH TARGET ACCURACY

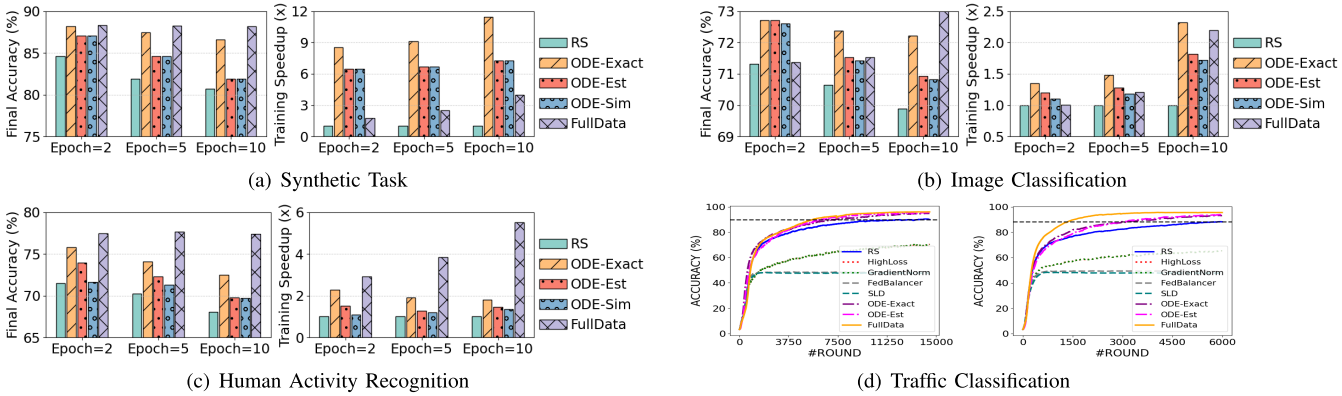
Properties	Final Accuracy (%)					
	RS	HL	GN	FB	SLD	ODE
Limited Storage	79.6	78.4	83.3	77.6	76.9	87.1
Streaming Data	79.6	87.2	87.6	87.2	87.0	88.2
Both Two	79.6	78.4	83.3	78.5	82.3	87.1
Properties	**Training Speedup (×)**					
Limited Storage	1.00	–	4.87	–	–	9.52
Streaming Data	1.00	2.86	2.30	2.86	2	2.67
Both Two	1.00	–	4.87	–	4.08	9.52

C. Overall Performance

We compare the performance of ODE with six baselines, and show the main results in Table IV.

TABLE IV
OVERALL PERFORMANCE OF DIFFERENT DATA SELECTION METHODS. SYMBOL '–' MEANS FAILING TO REACH THE TARGET ACCURACY

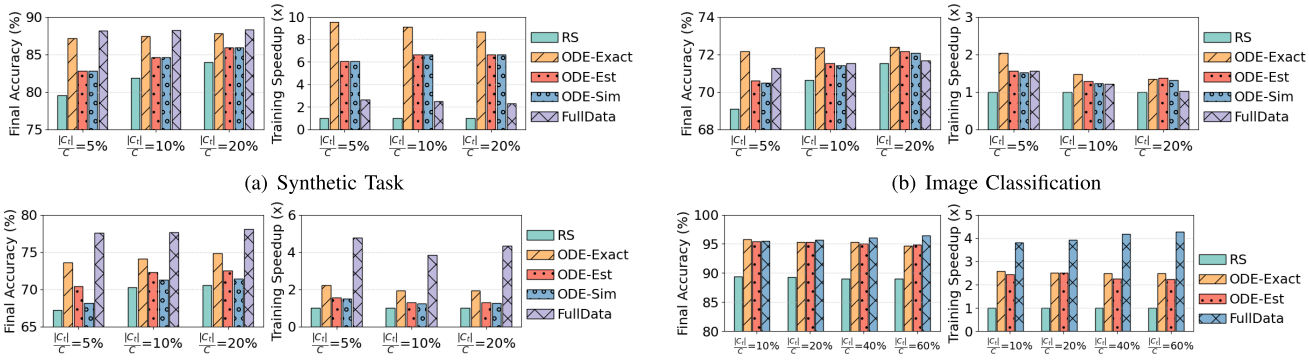
Task	Model Training Speedup (×)							
	RS	**HL**	**GN**	**FB**	**SLD**	**ODE-Exact**	**ODE-Est**	**FD**
ST	1.00	–	4.87	–	4.08	9.52	5.88	2.67
IC	1.00	–	–	–	–	1.35	1.20	1.01
HAR	1.00	–	–	–	–	2.22	1.55	4.76
TC	1.00	–	–	–	–	2.51	2.50	3.92
Task	**Final Inference Accuracy (%)**							
ST	79.6	78.4	83.3	78.5	82.4	87.1	82.8	88.1
IC	71.3	51.9	41.4	60.4	69.2	72.7	72.7	71.4
HAR	67.3	48.2	51.0	48.3	56.2	73.6	70.4	77.5
TC	89.3	69.0	69.3	72.2	72.3	95.3	95.3	96.0



(a) Synthetic Task (b) Image Classification

(c) Human Activity Recognition (d) Traffic Classification

Fig. 10. Robustness to number of local training epochs.



(a) Synthetic Task (b) Image Classification

(c) Human Activity Recognition (d) Traffic Classification

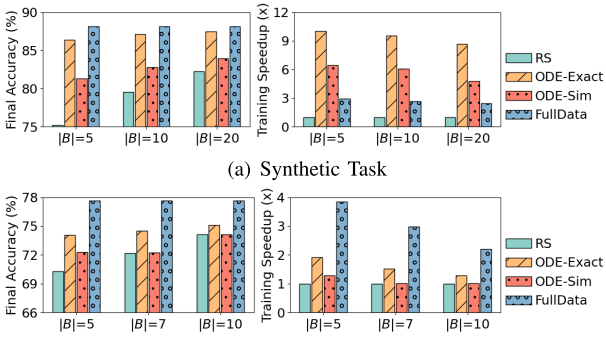
Fig. 11. Robustness of ODE to client participation rate.

ODE significantly speeds up the model training process on all datasets. Compared with baselines, ODE achieves the target accuracy $5.88-9.52\times$ faster on ST; $1.20-1.35\times$ faster on IC; $1.55-2.22\times$ faster on HAR; $2.5-2.51\times$ faster on TC. Also, we observe that the highest speedup is achieved on ST, because the high non-i.i.d degree across clients and large data divergence within each client leave a great potential for ODE to reduce data heterogeneity and improve training process.

ODE improves the inference accuracy of the final model. Table IV shows that compared with baselines, ODE enhances final accuracy on all the datasets, achieving $3.24-7.56\%$ higher on ST, $3.13-6.38\%$ on HAR, and around 6% on TC. We also notice that ODE has a marginal accuracy improvement ($\approx 1.4\%$) on IC, because the FashionMNIST dataset has less data variance within each label, and a randomly selected subset is sufficient to represent the entire data distribution.

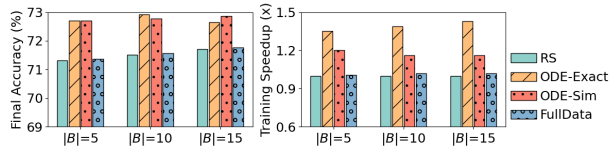
Importance-based data selection methods perform poorly. Table IV also reveals that these methods cannot reach the target final accuracy on tasks of IC, HAR and TC, as these real-world datasets contain noise, making such importance-based methods fail to work [13], [14].

Previous data selection methods for FL outperform importance-based methods but worse than ODE. As shown in Table IV, *FedBalancer* and *SLD* perform better than *HL* and *GN*, but worse than *RS* in a large degree, which is different from the phenomenon in traditional settings [13], [14], [34].

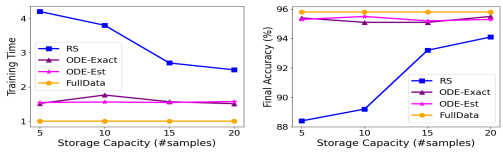


(a) Synthetic Task

(c) Human Activity Recognition



(b) Image Classification



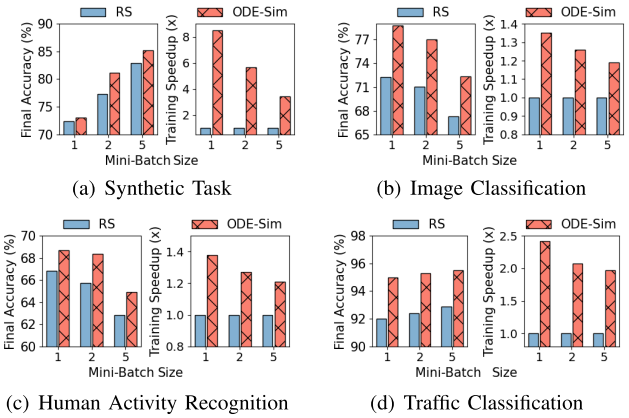
(d) Traffic Classification

Fig. 12. Robustness of ODE to on-device storage capacities.

TABLE V

MEMORY FOOTPRINT AND EVALUATION DELAY PER SAMPLE OF DIFFERENT DATA SELECTION METHODS ON THREE REAL-WORLD DATASETS

Task	Memory Footprint (MB)				
	RS	HL	GN	ODE -Est	ODE -Sim
IC	1.70	11.91	16.89	18.27	16.92
HAR	1.92	7.27	12.23	13.46	12.38
TC	0.75	10.58	19.65	25.15	14.47
Task	**Evaluation Time (ms per sample)**				
IC	0.05	11.1	21.1	22.8	11.4
HAR	0.05	0.36	1.04	1.93	0.53
TC	0.05	1.03	9.06	9.69	1.23



(a) Synthetic Task

(b) Image Classification

(c) Human Activity Recognition

(d) Traffic Classification

Fig. 13. Robustness of ODE-Sim to various mini-batch sizes.

This is because (i) their noise cleaning steps, such as removing samples with top loss or gradient norm, highly rely on the complete statistical information of full dataset, and (ii) their on-client data valuation metrics fail to work for the global model training in FL, as discussed in Section I.

Simplified ODE reduces computation and memory costs significantly with marginal performance degradation. We conduct another two experiments which utilize only the last 1 and 2 layers for data valuation on the industrial TC dataset. Empirical results shown in Figure 9 demonstrate that the simplified version reduces as high as 44% memory and 83% time delay, with only 0.1× and 1% degradation on training speedup and accuracy improvement. Unbalanced data sizes of clients.

ODE introduces small extra memory footprint and data evaluation delay. Empirical results in Table V demonstrate that

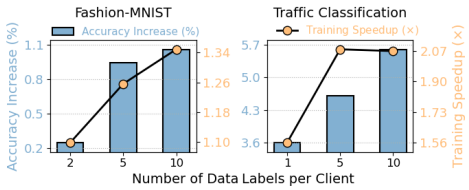


Fig. 14. Robustness of ODE-Sim to data heterogeneity.

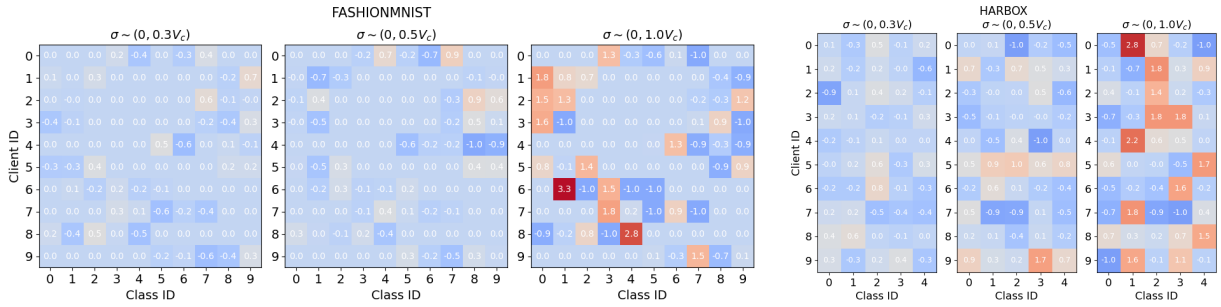
simplified ODE (*ODE-Sim*) brings only tiny evaluation delay and memory burden to devices (e.g., 1.23ms and 14.47MB for industrial TC task), and thus is applicable to practical network.

D. Robustness of ODE

In this section, we compare the robustness of ODE with baselines to various environmental factors, including the number of local training epoch, client participation rate, storage capacity, mini-batch size, data heterogeneity across clients, and the statistical error of clients' data velocities. For the three public datasets, we test the robustness of ODE-Exact, ODE-Est and ODE-Sim (using only last model layer), while for the industrial TC dataset, we only report the performance of ODE-Exact and ODE-Est, as ODE-Sim has quite similar performance with ODE-Est.

Number of Local Training Epoch. Empirical results shown in Figures 10 demonstrate that (i) ODE can work with various local training epoch numbers m. With m increasing from 2 to 10, all versions of ODE achieve higher training speedup and final inference accuracy than existing methods; (ii) ODE-Est has similar performance with ODE-Exact; (iii) ODE-Sim has slightly inferior performance compared with ODE-Est, but consistently outperforms *RS*, which inspires us to trade-off the efficiency and effectiveness of ODE in practical deployment by adjusting the number of model layers for data selection. The first phenomenon coincides with the previous analysis in Section III-A: (i) for convergence rate, the one-step-look-ahead strategy of ODE only optimizes the loss reduction of the first local training epoch in Eq. (3), and thus the influence on model convergence will be weakened when the number of local epoch increases; (ii) for inference accuracy, ODE narrows the gap between the models trained through FL and CL by optimizing the dominant term with the maximum weight of the gap bound in (5), leading to better final inference accuracy with a larger m.

Participation Rate. The results in Figure 11 demonstrate that all versions of ODE can improve the FL process



(a) Image Classification (b) Human Activity Recognition

Fig. 15. We visualize the relative observation error in data generation velocities of random 20 clients.

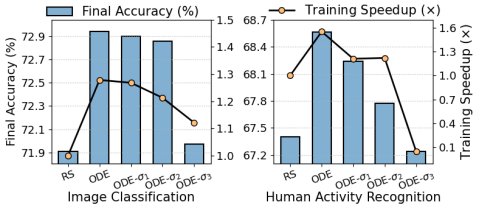


Fig. 16. Impact of clients' statistical errors in data generation velocities.

significantly even with small client participation rate. Specifically, ODE−Exact accelerates the model training by 2.57× and increases the inference accuracy by 6.6% in TC task, 9× and 8% in ST, 2× and 3% in IC, and 2× and 6% in HAR. ODE−Est achieves 2.4× training speedup and 6.1% accuracy increase in TC, 6× and 4% in ST, 1.5× and 2% in IC, 1.3× and 4% in HAR. Furthermore, we observe that ODE−Sim performs slightly worse than ODE−Est, due to the inconsistent data values computed from all model layers and only final layer. This demonstrates the practicality of ODE in the real-world environments, where only a small proportion of devices are available to participate in each FL round. We also notice that in some tasks, the training speedup of ODE degenerates with high client participation rate, which weakens the impact of data heterogeneity across clients and the effectiveness of ODE

Storage Capacity. We conduct experiments on various storage capacities of devices, and plot the training time speedup and final accuracy of each data selection method in Figure 12, which demonstrate that (i) the performance of *RS* degrades rapidly with the decrease of storage capacity in most tasks; (ii) ODE has stable performance across different storage capacities, and thus is more robust to potentially diverse storage capabilities of heterogeneous devices; (iii) Compared with ODE, *RS* needs more than twice storage space to achieve the same model training performance.

Mini-batch Size. As mini-batch SGD is widely used in FL to accelerating model training, we evaluate ODE−Sim on various mini-batch sizes, where we adjust the learning rate from η to 0.2η to reduce the instability of local model training process. Experimental results in Figure 13 demonstrate that (i) ODE−Sim consistently improves the performance of FL across various mini-batch sizes and real-world FL tasks; (ii) The smaller batch size is, the more training speedup and accuracy improvement can be achieved by ODE−Sim, which coincides the results of different numbers of local training epochs.

Data Heterogeneity. Similar with previous work, we manually adjust the number of data labels assigned to each client to simulate different heterogeneity levels. In ST and HAR task, the data has been already distributed to clients during the data collection process, and thus we focus on IC and TC datasets. Empirical results in Figure 14 illustrate that ODE achieves $3.6-5.6\%$ increase in inference accuracy and $1.56-2.07\times$ speedup in training time in different heterogeneous settings of TC task, $0.25-1.02\%$ and $1.11-1.34\times$ in IC task. Despite that the performance of ODE seems to degrade in the extremely heterogeneous case (#Label=1), such degradation is caused by the limited number of data labels owned by each client, which diminishes the effectiveness of the cross-device data selection component of ODE. This situation typically does not appear in practice as real-world devices usually possess multiple data labels (*e.g.*, mobile applications).

Statistical Error in Data Generation Velocities. In real world, the statistical information collected by clients from previous time periods may not be accurate, which potentially deteriorates the effectiveness of cross-client coordination strategy and the overall performance of ODE. To test the impact of such issue, we introduce an statistical error $\sigma_c \in \mathbb{R}^{|Y|}$ to the data velocity matrix V_c of each client c, where the error σ_c is assumed to follow a Gaussian distribution with zero mean. We conduct experiments on two real-world FL tasks (IC and HAR) with three distinct levels of statistical error: $\sigma_c \sim \mathcal{N}(0, 0.3V_c), \mathcal{N}(0, 0.5V_c), \mathcal{N}(0, 1.0V_c)$. The relative error rates of clients and data labels are visualized in Figure 15, and experimental results in Figure 16 shed light on two key observations: (i) As the error level rises, both the final inference accuracy and model training speedup achieved by ODE declines. (ii) ODE fails to be effective with the highest error level $\sigma_c \sim \mathcal{N}(0, V_c)$. Such outcome is acceptable, as this level introduces deviations as high as 330% from the true data generation velocity of each client and label, which is relatively rare for practical devices with the help of historical data.

E. Component-Wise Analysis

In this section, we evaluate the effectiveness of each key component in ODE: on-device data selection, global gradient estimator and cross-client coordination algorithm.

On-Client Data Selection. To show the effect of on-client data selection, we compare ODE with *Valuation-*, which replaces our on-device data selection method with *RS*, but still allows the server to coordinate the clients for collaborative

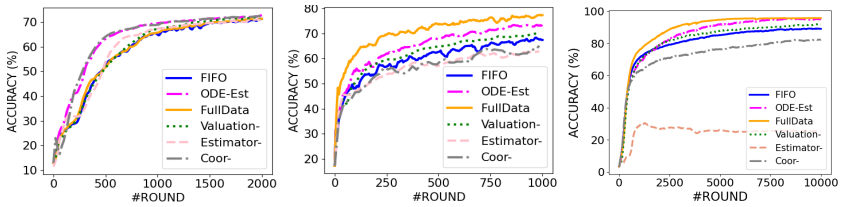


Fig. 17. Component-wise analysis of ODE. Left: Image Classification. Middle: Human Activity Recognition. Right: Traffic Classification.

data storage. Figure 17 shows that *Valuation-* performs slightly better than *RS* but much worse than *ODE-Est*, which implies the significant role of our proposed data valuation metric and data selection policy.

Global Gradient Estimator. To show the effectiveness of the global gradient estimator, we compare *ODE-Est* with a naive estimation method, namely *Estimator-*, where server only aggregates the local gradient estimators of participating clients to obtain a global gradient estimator, and each participant only leverages the locally stored data instead of all generated data to compute the local gradient. Figure 17 shows that the naive estimation method has the poorest performance, as the partial clients and biased local gradient lead to inaccurate estimation for global gradient, misleading clients to select data samples using a wrong data valuation metric.

Cross-Client Coordination. We conduct another experiment where clients select and store data samples only based on their local information without the coordination of the server, namely *Coor-*. Figure 17 shows that the performance of ODE is largely weakened in nearly all datasets, as the clients tend to store similar and overlapped valuable data samples, and thus other data will be under-represented. However, textitCoor- has similar performance with ODE in IC task, because the number of data labels owned by each client is the same as the number of labels assigned to each client, diminishing the effectiveness of cross-client collaborate data storage.

V. Related Works

Federated Learning is a distributed learning framework aiming to learn a global model over multiple devices' data without data sharing [9], [12]. Existing works mostly focus on how to improve FL from the perspectives of saving communication and computation costs. Previous works on communication cost reduction mainly leverage model quantization [35], sparsification [36] and pruning [37] to reduce the size of parameters transmitted between clients and server, or consider hierarchical FL structure to improve communication efficiency [38], [39]. Other literature enhances computation efficiency of FL [40] from different perspectives, such as improving the algorithms of local model update [41] and global model aggregation [23], [24], optimizing the client selection policy to mitigate the heterogeneity across clients [42], [43], [44], and etc. However, most of them assume a static training dataset on each device and overlook the practical properties of limited on-device storage and streaming networked data, which hinder the successul application of FL to mobile networks.

Data Selection. In FL, selecting data from streaming data can be seen as selecting batches of data from the underlying data distribution. To improve the model training process, existing methods select the most important data to participate in model update through different metrics. Leave-one-out test [15] and Data Shapley [16] quantify the value of each data sample as its marginal contribution to the model performance, such as the testing accuracy reduction or testing loss increase when removing the data sample from the training dataset. However, these approaches require multiple times of model retraining to obtain the accurate performance of models when removing certain data. Importance sampling [34] evaluates data sample through training loss or gradient norm over current model, which have been demonstrated to be theoretically optimal for mini-batch SGD. Other heuristic data selection metrics include data representativeness [45], model uncertainty [46], and etc. However, previous work [13], [14] on data selection in FL simply execute the above data selection process on each client for local training speedup without considering the impacts on global model training. Further, all of them require either access to complete dataset or multiple inspections over the streaming data, which is not practical in mobile networks.

Traffic Classification is a significant task in mobile networks, which associates traffic packets with specific applications for the downstream network management task, such as capacity planning and resource provisioning [47]. ML makes it possible to directly input the raw features of packet data into models for application identification or traffic classification, eliminating the need for manual feature extraction. Typical raw features can be categorized into three types. (i) Ports: the ports in the TCP/UDP header of some packets are associated with the well-known port numbers assigned by IANA,[3] which can server as the key features in determining the source application of packet. (ii) Raw Bytes: The actual flow bytes from packet headers and payloads can be leveraged for traffic classification, as they contain the raw content transmitted by the packet and thus can provide valuable information for identifying the packet's purpose [48]. However, this approach introduces high computational overhead and is not suitable for encrypted packet data [49]. (iii) Statistics: Statistical information includes mean, standard, deviation, minimum, maximum, of packet lengths, inter-arrival times, flow durations, number of packets, number of bytes, etc. These statistics form a feature vector for each flow and are widely employed to overcome the challenges posed by encrypted payload and users' privacy concern.

VI. Conclusion and Future Work

In this work, we identify two key properties of networked FL in mobile networks: limited on-device storage and

[3]https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

streaming networked data, which have not been explored in previous literature. Then, we present the design, implementation and evaluation of ODE, which is an online data selection framework for FL with limited on-device storage and consists of two components: on-device data evaluation and cross-device collaborative data storage. Extensive experiments on three public and one industrial dataset demonstrate that ODE significantly outperforms state-of-the-art data selection methods in terms of training time, final accuracy and robustness to various environmental factors.

In our research, we primarily focus on FL tasks involving automatic data labeling and stable data distribution of each client, which represent a wide range of mobile applications. For example, in keyboard prediction, a mobile user typically types $2,000$ characters per day, which can naturally serve as labels for preceding words. In mobile traffic classification, smart home routers can receive over $5,000$ non-encrypted packets per hour, which can be directly analyzed and labeled [47]. In image classification, the photo tags uploaded or corrected by users can be regarded as the image labels. In these applications, the interests and behavior of mobile users tend to remain stable over a certain period, indicating that the data distribution of each client typically does not undergo significant changes. Also, the utilizing unlabeled and dynamic streaming data is an important but under-explored area, which will be investigated in our future work.

ACKNOWLEDGMENT

The opinions, findings, conclusions, and recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

REFERENCES

[1] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, Feb. 2020.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[3] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 1048–1056, Aug. 1995.

[4] M. Siddiqi, H. Yu, and J. Joung, "5G ultra-reliable low-latency communication implementation challenges and operational issues with IoT devices," *Electronics*, vol. 8, no. 9, p. 981, Sep. 2019.

[5] P. K. Aggarwal, P. Jain, J. Mehta, R. Garg, K. Makar, and P. Chaudhary, "Machine learning, data mining, and big data analytics for 5G-enabled IoT," in *Blockchain for 5G-Enabled IoT*. New York, NY, USA: Springer, 2021, pp. 351–375.

[6] D. Rafique and L. Velasco, "Machine learning for network automation: Overview, architecture, and applications," *J. Opt. Commun. Netw.*, vol. 10, no. 10, pp. D126–D143, 2018.

[7] Y. Deng, "Deep learning on mobile devices: A review," in *Mobile Multimedia/Image Processing, Security, and Applications*. Bellingham, WA, USA: SPIE, 2019.

[8] X. Zeng, M. Yan, and M. Zhang, "Mercury: Efficient on-device distributed DNN training via stochastic importance sampling," in *Proc. 19th ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2021, pp. 29–41.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2017, pp. 1273–1282.

[10] UCSC. (2020). *Packet Buffers*. [Online]. Available: https://people.ucsc.edu/~warner/buffer.html

[11] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5132–5143.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[13] J. Shin, Y. Li, Y. Liu, and S. Lee, "Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2022, pp. 436–449.

[14] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[15] R. D. Cook, "Detection of influential observation in linear regression," *Technometrics*, vol. 19, no. 1, p. 15, Feb. 1977.

[16] A. Ghorbani and J. Zou, "Data Shapley: Equitable valuation of data for machine learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2242–2251.

[17] P. Zhao and T. Zhang, "Stochastic optimization with importance sampling for regularized loss minimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1–9.

[18] S. Caldas et al., "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*.

[19] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[20] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "ClusterFL: A similarity-aware federated learning system for human activity recognition," in *Proc. 19th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2021, pp. 54–66.

[21] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, 1985.

[22] L. Shapley, *Quota Solutions Op N-person Games1*, E. Artin and M. Morse, Eds. Princeton, NJ, USA: Princeton Univ. Press, 1953, p. 343.

[23] S. Zawad et al., "Curse or redemption? How data heterogeneity affects the robustness of federated learning," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2021, pp. 10807–10814.

[24] Z. Chai et al., "Towards taming the resource and data heterogeneity in federated learning," in *Proc. USENIX Conf. Oper. Mach. Learn. (OpML)*, 2019, pp. 19–21.

[25] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.

[26] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput. (ICTAMC)*, 2008, pp. 1–19.

[27] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of L_P minimization," *Math. Program.*, vol. 129, no. 2, pp. 285–299, Oct. 2011.

[28] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for L1 regularization: A comparative study and two new approaches," in *Proc. Eur. Conf. Mach. Learn. (ECML)*, 2007, pp. 286–297.

[29] S. Hong, S. Kim, J. Choi, Y. Lee, and J. H. Cheon, "Efficient sorting of homomorphic encrypted data with K-way sorting network," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4389–4404, 2021.

[30] Y. LeCun et al., "Handwritten digit recognition with a back-propagation network," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 1989, pp. 396–404.

[31] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," 2015, *arXiv:1511.06343*.

[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–21.

[33] T. B. Johnson and C. Guestrin, "Training deep models faster with robust, approximate importance sampling," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 7276–7286.

[34] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 2525–2534.

[35] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2021, pp. 2350–2358.

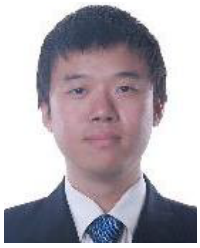
[36] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 300–310.

[37] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: An efficient federated learning framework for heterogeneous mobile clients," in *Proc. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2021, pp. 420–437.

[38] S. Hosseinalipour et al., "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1569–1584, Aug. 2022.
[39] X. Liu et al., "Accelerating federated learning via parallel servers: A theoretically guaranteed approach," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2201–2215, Oct. 2022.
[40] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.
[41] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.
[42] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2022, pp. 10351–10375.
[43] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. ICC IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[44] F. Li, J. Liu, and B. Ji, "Federated learning with fair worker selection: A multi-round submodular maximization approach," in *Proc. IEEE Int. Conf. Mobile Ad-Hoc Smart Syst. (MASS)*, 2021, pp. 180–188.
[45] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 6950–6960.
[46] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1002–1012.
[47] I. Akbari et al., "A look behind the curtain: Traffic classification in an increasingly encrypted web," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, pp. 1–26, Feb. 2021.
[48] Y. Wang, Y. Xiang, and S.-Z. Yu, "Automatic application signature construction from unknown traffic," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 1115–1120.
[49] M. Finsterbusch, C. Richter, E. Rocha, J.-A. M'uller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, 2nd Quart., 2013.

Chen Gong (Student Member, IEEE) received the B.E. degree in computer science from Shanghai Jiao Tong University in 2022, where he is currently pursuing the Ph.D. degree in computer science and technology. His research interests include mobile computing and on-device machine learning.

Zhenzhe Zheng (Member, IEEE) received the B.E. degree in software engineering from Xidian University in 2012 and the M.S. and Ph.D. degrees in computer science and engineering from Shanghai Jiao Tong University in 2015 and 2018, respectively. He is currently an Associate Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has visited the University of Illinois at Urbana–Champaign (UIUC) as a Visiting Scholar and then a Post-Doctoral Research Associate from 2016 to 2019. His research interests include intelligent mobile computing and large-scale decision-making. He was a recipient of NSFC Excellent Young Scholars Program, CCF-Intel Young Faculty Researcher Program Award, and China Computer Federation (CCF) Excellent Doctoral Dissertation Award. He has served as a member of Technical Program Committee for several academic conferences, such as INFOCOM, MobiHoc, KDD, WWW, AAAI, and IoTDI. He is a member of the ACM and CCF. For more information visit the link (https://zhengzhenzhe220.github.io/).



Yunfeng Shao received the B.S. degree in electronic engineering from Shanghai Jiao Tong University in 2009 and the M.S. degree from the University of Chinese Academy of Sciences China, in 2014. He is currently an Expert with the Huawei Noah's Ark Laboratory. He has published multiple papers at top-tier conferences, including NeurIPS, ICML, KDD, and PMLR. His research interests include retrieval-based language models, machine learning with privacy protection, and federated learning and their applications.

Bingshuai Li received the B.S. and M.S. degrees from Jilin University, China, in 2014 and 2017, respectively. He is currently a Senior Engineer with the Huawei Noah's Ark Laboratory. His current research interests include retrieval-based language models, machine learning, federated learning, transfer learning, and their applications in telecommunication networks.

Fan Wu (Member, IEEE) received the B.S. degree in computer science from Nanjing University in 2004 and the Ph.D. degree in computer science and engineering from the State University of New York at Buffalo in 2009. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has visited the University of Illinois at Urbana–Champaign (UIUC) as a Post-Doctoral Research Associate. His research interests include wireless networking and mobile computing, data management, algorithmic network economics, and privacy preservation. He has published more than 200 peer-reviewed papers in technical journals and conference proceedings. He was a recipient of the First Class Prize for Natural Science Award of China Ministry of Education, China National Fund for Distinguished Young Scientists, ACM China Rising Star Award, CCFTencent "Rhinoceros Bird" Outstanding Award, and CCF-Intel Young Faculty Researcher Program Award. He has served as an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING and *ACM Transactions on Sensor Networks*, an Area Editor for *Computer Networks* (Elsevier), and a member of technical program committees for more than 100 academic conferences. For more information visit the link (http://www.cs.sjtu.edu.cn/ fwu/).

Guihai Chen (Fellow, IEEE) received the B.S. degree from Nanjing University in 1984, the M.E. degree from Southeast University in 1987, and the Ph.D. degree from The University of Hong Kong in 1997. He is currently a Distinguished Professor with Shanghai Jiao Tong University, China. He had been invited as a Visiting Professor by many universities, including Kyushu Institute of Technology, Japan, in 1998, The University of Queensland, Australia, in 2000, and Wayne State University, USA, from September 2001 to August 2003. He has a wide range of research interests, with a focus on sensor networks, peer-to peer computing, high-performance computer architecture, and combinatorics. He has published more than 200 peer-reviewed articles and more than 120 of them are in well-archived international journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Journal of Parallel and Distributed Computing*, *Wireless Networks*, *The Computer Journal*, *International Journal of Foundations of Computer Science*, and *Performance Evaluation*; and also in well-known conference proceedings, such as HPCA, MOBIHOC, INFOCOM, ICNP, ICPP, IPDPS, and ICDCS.